Journal of Computer and Knowledge Engineering, Vol. 1, No. 1
DOI: 10.22067/cke.v1i2.53554

Scheduling Data-Driven Workflows in Multi-Cloud Environment

Nafise Sooezi[✦], Saeid Abrishami, Majid Lotfian

Abstract. Nowadays, cloud computing and other distributed computing systems have been developed to support various types of workflows in applications. Due to the restrictions onthe use ofone cloud provider, the concept of multiple clouds as been proposed.Inmultipleclouds, schedulingworkflowswithlarge amounts ofdata is a well-knownNP-Hard problem. The existing scheduling algorithms have not paid attention to the data dependency issues and their importance in scheduling criteria such as time and cost. In this paper, we propose a communication-based algorithm for workflows with huge volumes of data in a multi-cloud environment. The proposed algorithm changes the definition of the Partial Critical Paths(PCP) to minimize the cost of workflow executionwhile meeting a user defined deadline.

Keywords: **C**loud Computing,Multi-cloud,Workflow Scheduling,Data Dependency, Communication.

1. Introduction

Multiple clouds have a special place in the modern-day models we use. An important reason for this is the increased use of clouds in recent years. One of the important features of clouds is the illusion of unlimited resources to cloud users. The number of users varies at different times of the day during the weeks or on weekends. If the providers upgrade their resources so as to meet the peak demand of users, these resources will remain partially unused during non-peak hours. However, providers can shut down unused nodes in order to eliminate the cost of maintenance of the equipment and resources; however, they still have to pay for the cost of buying and equipping these unused resources. To offset these costs, providers are forced to increase their prices, and this has resulted in the poor competition between cloud service markets. On the other hand, if the provider only supplies the needed resources to users in the average demand time, then it cannot provide service at peak time demands and this will lower the reliability of the provider and it will result in a reduction of the number of users of its services. Today, a cloud alone cannot meet the needs of users at all times and it is becoming more important to provide service using multiple clouds. Sharing resources between several providers might be the best solution to the problem.

Manuscript received February 7, 2016; revised July 18, 2016; accepted November 24, 2016.
Nafise Sooezi and Saeid Abrishami, Department of Computer Engineering, Engineering Faculty, Ferdowsi University of Mashhad, Mashhad, Iran.
Majid Lotfian, Department of Computer Engineering, Engineering Faculty, Islamic Azad University of Mashhad, Mashhad, Iran.
[✦]The corresponding author's e-mail is: nafise.sooezi@stu.um.ac.ir

In [1], more than 25 types of multiple clouds have been introduced among which multi-cloud and cloud federation are a few examples. In cloud federation, cloud providers will agree to share resources, which help them improve services to the users. However, since the cloud technology is in its early stages and there is no overall standard, the agreement between cloud providers is difficult due to the fact that each provider uses their own interface and protocol. Multi-clouds a type of multiple cloud system in which there is no agreement between the cloud providers, and a third party is responsible for the relationships, the dialogues, and monitoring the providers. Scheduling in a cloud environment has been one of the major challenges in the world of clouds.

Workflow is a collection of interconnected multiple tasks that must be performed in a specific order. Workflow structure indicates temporary dependencies between tasks [2]. Workflow scheduling is the problem of mapping each task to a suitable resource and of ordering the tasks on each resource to satisfy some performance criterion. This is an NP-Hard problem, so there is no known polynomial algorithm for it. In general, Multi-Criteria scheduling problems are very difficult to solve even in the single cloud case. Workflow scheduling is facing more challenges in the multi-cloud environment due to the increasing number of complex factors. One of the major problems facing the proposed scheduling algorithms in multiple clouds is the lack of attention to communication in workflow and its effect on the cost and execution time. The data transfer rate between the samples of a cloud is very high, (e.g. bandwidth between different samples in the Amazon EC2 is approximately between 300 Mbps to 4 Gbps) and this transfer is free, while the received speed of the Amazon cloud (inter-cloud speed) is between 400 Mbps to 20 Mbps and send speed is between 20Mbps and 80 Mbps[3-5]. In the proposed solution, we have tried to pay attention to minimize the cost and time due to it. In the proposed approach, we use the concept of Partial Critical Paths(PCP) introduced by Abrishami et al.[6]. In this way, we have changed the definition of the critical path and the amount of communication between the tasks to be included in this definition.

The rest of the paper is organized as follows. Section 2 presents related work. In section 3, we describe the generic application, objective, and platform models underneath our approach. Section 4 shows the proposed algorithm and scheduling policies. Section 5 presents and discusses the results. Finally, in Section 6 we present our conclusions and future work.

2. Related Work

So far, many algorithms have been proposed for workflow scheduling in a single cloud including: [7], [8], [6], [9], [10] and [11].The authors of most of these works have considered execution time and cost as their objectives. Ever since the concept of multi-cloud has been introduced in recent years, there have been a few algorithms in this field.

In 2007, Sakellariou et al. [12] presented an algorithm known as IC-LOSS. The IC-LOSS algorithm tries to minimize the execution time under a budget constraint. This algorithm consists of two phases: first, it tries to find an initial schedule for the input workflow with minimum execution time, and then it refines the initial schedule until its budget constraint is satisfied. In the first phase, it uses a well-known makespan minimization algorithm, called HEFT [13]. The HEFT algorithm for each task is looking for a version that has the earliest completion time for the task. The second part deals with the correction of the scheduling with the allocation of tasks to cheaper resources until the budget constraints are fulfilled:

LossWeight $(T_I, J) = \frac{\text{Tnew} - \text{Told}}{\text{Cold} - \text{Cnew}}$

In 2010, Van den Bossche et al.[14]solved the problem of multiple applications scheduling in several cloud providers using linear programming. The objective function in this algorithm is minimizing the total cost of data traffic and computational cost over all time slots within the schedule for all providers. The lack of consideration of the communication time between the clouds is one of the problems in this paper.

In 2011, Houidi et al. [15] presented an algorithm that aims to break several requests between cloud providers such that the user costs (total cost of each resource and the cost of communication) are minimal. They distributed the requests with the broker between the cloud providers. The broker is composed of three main components:

1. Cloud Request Splitting
2. Resource Provisioning
3. Inter-Cloud Network Provisioning.

After formulating the problem, they have solved it by using linear programming.

In 2011, Li et al. [16] presented an algorithm that aims to maximize capacity and minimize costs in accordance with the new conditions of the providers. They have also tried to minimize the overhead of scheduling under the new conditions as compared with the previous condition if some changes are made in the environment.

In 2013, Fard et al. [17] presented a method to prevent the selfish behavior of providers that use an auction pricing model instead of the pay as you go model. In this way, each task announces to the resources its workload (communicating with other tasks and the required input-output). The source suggests an approach to tasks. In this way, the solution is chosen so that the product of time and its cost is minimal. After winner resource is selected, if the time proposed by the source is greater than or equal to real-time, the cost of the provider is fully paid and if the time proposed by the source is less than real time, the resource is penalized using a given function. In this method, the Nash equilibrium is used that is a fundamental concept of the theory of games.

In 2012, Fardet al. [18] presented an algorithm that is one of the complete algorithms introduced in this field. The algorithm makes use of user-defined constraints about time, cost, power consumption, and reliability and then it estimates the optimal solution. In this paper, all the objectives are modeled. Then the algorithm approximates the optimum solution during threephases. In the first phase, it estimates the objectives' sub-constraints for each individual task using the user constraint vector. In the second phase, it assigns a rank to each task of the workflow and sorts them in an ascending order. Finally, in the third phase, the algorithm attempts to allocate the most appropriate resource to each activity with due consideration given to the estimated sub-constraints. A major problem with this algorithm is that it does nothing to improve communication. As was mentioned earlier in this paper, inter-cloud communication is one of the most important issues in the scheduling workflow in multi-cloud systems. Lack of attention to this point has affected the whole algorithm, and it is particularly inappropriate for communication-based workflows. The assumption of unlimited resources is another problem in this algorithm.

Duan et al. [19] offered a good algorithm in 2014. In their paper, time and cost are considered based on the limitations of communication bandwidth and storage space. One of the differences between this paper and the previous one [18]is that this paper considers two objectives and two conditions instead of four objectives. The other difference is that this algorithm has a lower time complexity as compared with [18]. In this paper, the problem is modeled with game theory. The algorithm is repeated as many times as needed by one condition and the nearly optimal solution is found. One of the advantages of this algorithms fasts convergence by using the information about the environment and the competitors. And the other advantage is that you can easily add a new objective to the problem. One of the major problems of this algorithm is that it is not suitable for applications with a high level of complexity just like the algorithm presented in the previous paper. Other problems can also be mentioned such as the following:

1. Initialization of the weight vector is done by the algorithm itself and this can lead to different results.
2. Tasks are broken vertically to transfer parallel tasks to one provider that cannot be useful because there is no data to transfer between them (The cost and time of data transmission within a provider are not comparable to the inter providers).

In 2014, Montes et al. [20] proposed an algorithm that allows execution of dynamic workflows in a multi-cloud environment. In addition, there is an ability to customize the scheduler for the user. One of the policies that provide this ability operates as follow: it assigns instances to each task so that the total execution time of tasks, task data receiving time to the desired instance, and the estimated time needed to perform the next task, is minimized. Another policy is based on a deadline that selects a minimum set of the resources that are needed to complete all tasks such as deadlines are met and the objective function is satisfied. They have considered four objective functions: performance optimization, data locality optimization, performance and data optimization, and cost optimization. One of the major problems of this algorithm is the lack of attention to the communication problems costs. As the article mentioned, communication has a very great impact on the cost and time and that should be focused on. One of the other problems is considering the workflow dynamically and separating its steps effectively.

Table 1. Comparison of different scheduling algorithms in a cloud environment

Paper	*Environment*	*Scheduling type*	*Method*	*Objectives*	*Advantages*	*Disadvantages*
BIP [14] 2010	Hybrid cloud and Multi-cloud	static	Mathematical model	• Minimize cost	-	• The lack of attention to communication and to be placed tasks in a single provider • Unsuitable for communication-based applications
[15] 2011	Multiple cloud	static	Mathematical model	• Minimize cost	•consider tasks placed in a single provider	• The lack of attention to communication and execution time and user deadline
[16] 2011	Multiple cloud	dynamic	Mathematical model	• Maximize the use of capacity • Minimize cost in new conditions	• Dynamic considering the price of instances, the type of instances, and the performance of the services	• The lack of attention to be placed tasks in a single provider and communication in workflows
[17] 2013	Multi-cloud	dynamic	Mathematical Model	• Minimize cost • Minimize execution time	• harness the selfish behavior of cloud providers	• The lack of attention to be placed tasks in a single provider and communication in workflows
[19] 2014	Hybrid cloud	static	Mathematical Model	• Minimize cost & time while fulfilling network bandwidth and storage requirements	• fast convergence by using competitors and environment information	• unsuitable for applications with the complex dependencies between tasks • initialize the weight vector by the algorithm itself • break tasks vertically
MOLS [18] 2012	Set of heterogeneous resources	static	Heuristic	• Minimize cost, time, and energy consumption • Maximize reliability	• low time complexity	• assuming resources are unlimited • The lack of attention to communication between the clouds
[20] 2014	Multi-cloud	Dynamic	Heuristic	• Minimize cost to satisfy the objective function and user deadline	• allows users to customize scheduling policies	• The lack of attention to communication and cost at the same time • high time complexity

3. The Model

3.1. The Application Model

Workflow is described by a Directed Acyclic Graph (DAG) in which each computational task is represented by a node, and each data or control dependency between tasks is represented by a directed edge between the corresponding nodes. W=(T,E) consists of a set of *n* tasks: T= $\bigcup_{i=1}^{n} T_i$ interconnected through a set of control flow and data flow dependencies: E=$\{(T_i, T_j, Data_{ij}) | (T_i, T_j) \in T \times T\}$ As $Data_{ij}$ shows the amount of data to be exchanged between T_i and T_j.

We always add two dummy tasks T_{entry} and T_{exit} to the beginning and the end of the workflow, respectively. These dummy tasks have zero execution time and they are connected with zero-weight dependencies to the actual entry and exit tasks.

3.2. The Platform Model

A multi-cloud environment includes N providers: P_1, P_2,…,P_N. Each provider has certain characteristics that are shown by a property vector (B_{up}, B_{down}, C_{in}, C_{out}, $B_{internal}$, I), which include (in order) upload/download bandwidth, incoming/outgoing data transfer costs, internal bandwidth and set of provider's instances (I_{1i},I_{2i}, … I_{mi}). Each instance mi, has certain characteristics that are shown by a property vector $I_{mi}=(V_{mi},C_{mi})$, which include (in order) computational speed of the instance I_{mi} in millions of instructions per second (MIPS) and cost of instance *mi*.

3.3. The Objective Model

We want to schedule workflow so that the execution costs are minimized and user deadlines are satisfied. Time and cost have been formulated according to [18]. The execution time of task T_j on the instance I_{mi}can be computed as the sum of the longest input transfer time T_j (from all inputs to T_j) and the task computation time:

$$ET(T_j,I_{mi}) = \text{Max}_{\,T_P \in \text{pred}(T_j)} \left\{\frac{\text{data}_{pj}}{B_{up}(mi)}\right\} + \frac{\text{work}(T_j)}{V_{mi}} \qquad (1)$$

Where$B_{up}(mi)$is the bandwidth between task$T_j$and T_p. The completion time or makespan of a task $T_j$can be recursively computed as follows:

$$ET_{final}(T_j,I_{mi}) = \begin{cases} ET(T_j, I_{mi}) & \text{pred}(T_j) = \emptyset \\ \text{Max}_{\,T_p \in \text{pred}(T_j)} \left\{ET_{final}\left(T_p, \text{sched}(T_p)\right) + ET(T_j, I_{mi})\right\} & \text{pred}(T_j) \neq \emptyset \end{cases} \qquad (2)$$

Consequently, the workflow makespan is given by the longest completion time of its tasks:

$$Total_{ET}(workflow) = \text{Max}_{j \in [1 \dots n]} \{ET_{final}(T_j, \text{sched}(T_j))\} \qquad (3)$$

The cost of task $T_j$in the instance mi is thesum of the computation and data transfer costs:

$C(T_j,I_{mi})$
$=ET(T_j,I_{mi})*C_{mi}+Input(T_j)*C_{in_{mi}}+Output(T_j)*C_{out_{mi}}$(4)
where $C_{mi}$is the computational cost in instance mi, $C_{in_{mi}}$and $C_{out_{mi}}$ is the incoming and outgoing data transfer cost of instance I_{mi}'s cloud. Input (T_j) and Output (T_j) are the total amount of input and output data of node T_j that are received from tasks that have been scheduled in instances other than cloud of instance I_{mi}(because data transfer cost between tasks is zero if you have two tasks on the same instance or on instances that are available on one cloud). The execution costs for a workflow is equal to the sum of computation and communication costs for all tasks:

$$C_{final}(workflow)=\sum_{j=1}^{n} C(T_j, sched(T_j)) \quad (5)$$

4. The Proposed Algorithm

At first, a brief look at basic concepts is presented. In the proposed scheduling algorithms, we have two notions of the start times of tasks, i.e. the earliest start time computed before scheduling the workflow, and the actual start time which is computed after the tasks are scheduled. The Earliest Start Time of each unscheduled taskT_i, EST(T_i), is defined as follows:

$$EST(T_{entry}) = 0$$
$$EST(T_i)=Max_{T_p\in pred(T_i)}\{EST(T_p)+MET(T_p)+TT(T_p,T_i)\} \quad (6)$$

where the Minimum Execution Time of a task T_p, MET(T_p), is the execution time of task T_p on an instance $I_j \in I$ which has the minimum ET(T_p, I_j) between all available instances. Note that MET(T_{entry}) and MET(T_{exit}) equal zero.TT(T_p,T_i) is the data transfer time of the dependency $Data_{pi}$. Accordingly, the Earliest Finish Time of an unscheduled task T_i, EFT (T_i), can be defined as follows:

$$EFT (T_i) = EST (T_i) + MET (T_i) \quad (7)$$

The latest finish time for each unscheduled task is calculated as follows:

$$LFT(T_{exit}) = D$$
$$LFT(T_i) =Min_{T_p\in child(T_i)}LFT(T_p)—MET(T_p) —TT(T_p,T_i)\} \quad (8)$$

where LFT(T_i) is the latest time at which T_i can finish its computation such that the whole workflow can finish before the user defined deadline.

The general idea is that the proposed algorithm breaks workflow in such a way that tasks with the most dependency are scheduled to run on one cloud.

In the proposed algorithm, the critical paths are identified according to the new definition. EFT,EST, and LFT are computed for all nodes. Then we define the degree of dependence that is calculated for all nodes and finally for all paths. Nodes are ranked and scheduled so that the best possible cloud is assigned to each path. In the following, we explain the steps of the proposed algorithm.

4.1. Step One: Identify the Partial Paths with Minimal Communications

The Critical Parent of a node T_i is the unassigned parent of Ti that has the latest data arrival time at T_i. The partial critical path for each workflow graph is calculated as follows: we begin with T_{exit} and follow back the critical parents until we reach T_{entry}, and so we find the overall real critical path of the workflow graph .The proposed algorithm has changed this concept and the graph is broken into paths whose tasks together are the largest data exchange. One of the conditions of the generated paths is that all nodes have a maximum of one parent and one child in every path. Algorithm 1 shows how to break the workflow graph to paths with the mentioned conditions.

4.2. Step Two: Preprocessing

At this step, the Degree of Dependence (DOD) of each path to the other paths is calculated. Thus, because of the limitation of free capacity of every cloud, the algorithm specifies that the paths should be scheduled on one cloud. At first, DOD is calculated for tasks that their critical parents are located in another path according to Algorithm1. Thus, the DOD of T_1 to T_2 is equal to the start time of T_1 regarding the arrival time of data from T_2 in the other path, minus the start time of T_1 regardless of the arrival time of data from T_2 in the other path:

$$DOD(T_1,T_2) = (EST(T_2) + MET(T_2) + TT(T_1,T_2))-$$
$$(EST(T_3) + MET(T_3) + TT(T_1,T_3)) \quad (9)$$

Algorithm 1: Finding Critical Paths

```
1. Input:$W=(T,E),T=\bigcup_{i=1}^{n} T_i,E=\{(T_i ,T_j, Data_{ij})|(T_i,T_j)\in T \times T\}$,
   $E_{ij}=(T_i,T_j,data_{ij})$;
2. Output:CriticalPaths=$\bigcup_{i=1}^{s} CP_i$, $CP_i$=(startNode$_i$, endNode$_i$,$E'_i$| $E'_i$C E);
3. coveredPaths← $\emptyset$, coveredNodes← $\emptyset$;   /*set them to empty*/
4. SE← Sort(E,Data);  /*Sort all E in descending  Data order */
5. forall ($E_{ij}\in$ SE) do
6. if($E_{ij}\notin$coveredPathsand$T_i\in$cøveredNodesand$T_j\in$coy̸eredNodes)then
7.         add ($T_i$,$T_j$,$E_{ij}$) to coveredPaths
8.         add $T_i$,$T_j$ to coveredNodes
9.    end if
10. if($E_{ij}\notin$coveredPathsand$T_i\in$cøveredNodesand$T_j\in$coveredNodes)then
11.           if( $\exists CP_e\in$coveredPaths | $CP_e=(T_j,T_{j'},E_{jj'})$) then
12.           $CP_e=(T_i,T_{j'},E_{ij})$
13.                add $T_j$ to coveredNodes
14.           end if
15.    end if
16. if($E_{ij}\notin$coveredPathsand$T_i\in$coveredNodesand$T_j\in$cove̸redNodes)then
17.           if($\exists CP_e\in$coveredPaths |$CP_e=(T_{i'},T_i,E_{i'i})$) then
18.           $CP_e$= $(T_{i'},T_j,E_{ij})$
19.                 add $T_i$ to coveredNodes
20.           end if
21.    end if
22. if($E_{ij}\notin$coveredPathsand $T_i\in$coveredNodesand$T_j\in$coveredNodes)then
23.           if(($\exists CP_e\in$coveredPaths| $CP_e=(T_{i'},T_i,E_{i'i}$ ))and
              ($\exists CP_{e'}\in$coveredPaths | $CP_{e'}=(T_j,T_{j'},E_{jj'})$)) then
24.           $CP_e\leftarrow (T_{i'},T_{j'},E_{ij})$
25.                  Remove $CP_{e'}$fromcoveredPaths
26.           end if
27.    end if
28. end for
29. Return coveredPaths
```

whereT_1 is the critical parent of T_2 that has been located in a different path with the path of T_1 according to Algorithm 1, and T3 is the parent of T_1 in the path produced by Algorithm 1.It should be noted that TT(T_1, T_2) is calculated by using the inter-cloud speed while $TT(T_1,T_3)$ is calculated by using the intra-cloud speed. In addition, DOD is calculated only for two nodes such that one is the critical parent of the other, but has been located in a different path from the path produced by Algorithm 1, and DOD for other pairs of nodes is considered to be zero. After calculating the DOD of all nodes, DOD of each path to the other path is calculated, i.e. that is equal to the sum of DOD of all nodes in the path to nodes in the other path and vice versa:

$$DOD(CP_1,CP_2)=\sum_{T_i \in T'_{CP_1}} \sum_{T_j \in T'_{CP_2}} DOD(T_i,T_j) + \sum_{T_j \in T'_{CP_2}} \sum_{T_i \in T'_{CP_1}} DOD(T_j,T_i) \quad (10)$$

where T'_{CP_1} and T'_{CP_2} respectively are the set of nodes in the path CP_1 and CP_2. Algorithm 2 shows how to calculate DOD of two paths.

4.3. Step three: Allocation of Resources to the Partial Paths

At this step, a degree is assigned to each node in the workflow. The degree is based on the estimated time required to execute tasks. In this case, we begin with the end of the workflow and calculate the degree of root nodes. The degree is equal to the execution time of these tasks in the fastest available instance. Then, we calculate the degrees of all parent nodes. The degree of each parent is equal to the sum of the maximum degree of the children and data transmission time from the parent to the child. Similarly, the degrees of all tasks in the workflow are calculated. These degrees are arranged in ascending order. Then tasks from the ordered list are traversed and scheduled onto the best cloud. Algorithm 3 shows the procedure. In this algorithm, we begin from the node with the highest degree and based on the assigned degree, the operation is repeated for each node. The node that is to be scheduled is now called the current node. Among all the paths identified by Algorithm 1, the path that includes the current node is called the current path. First, we examine if there is a node on the current path that is scheduled. If the response is positive, we try to find the best instance preferably in the cloud in which the parent of the current node is placed(First we will search for available instances at the desired cloud, and if a good instance could not be found we create a new instance in that cloud). If the response is negative, we consider if there is a scheduled node on the paths that depends on the path of the current node. If affirmative, we try to find the best instance in the cloud on which the path that depends on to the current node is scheduled. If there are several options, we select a cloud that has a greater volume of transactions (according to the calculated DOD by using Algorithm 2) in the current path. If the response is negative, we introduce an appropriate cloud based on our preliminary estimates and create the best instance on it. The preliminary estimates check that if all workflow nodes are separately scheduled on one cloud, which cloud would be the least expensive one (Any node of the workflow that cannot be executed even in the fastest instance of a cloud is not considered). So, we select a cloud for which the ratio of the cost to the number of instructions of executable tasks is the least. After scheduling this node, the node with the next degree is chosen.

Algorithm 2: Computing Degree Of Dependency of two paths(DOD)

```
1. Input:W=(T,E), $T=\bigcup_{i=1}^{n} T_i$ , $E=\{(T_i,T_j, Data_{ij})|(T_i,T_j)\in T \times T\}$,
        $E_{ij}=(T_i,T_j,data_{ij})$, CriticalPaths=$\bigcup_{i=1}^{s} CP_i$;
2. Output:$\bigcup_{c=1}^{s(s-1)}\{DOD(CP_i,CP_j)|CP_i,CP_j \in$ CriticalPaths};
3. for all ($T_i \in T$) do
4.     Compute EST ($T_i$), EFT ($T_i$) and LFT ($T_i$);
5. end for
6. for all ((Ti , Tj)∈ (T × T)) do
7.  $CP_{T_i} \leftarrow$ FindCriticalPath($T_i$);    /*Path of $T_i$ form CriticalPaths*/
8. $CP_{T_j} \leftarrow$ FindCriticalPath($T_j$);    /*Path of $T_j$ form CriticalPaths*/
9. CiriticalParent$_{T_i} \leftarrow$ Max$_{T_p \in pred(T_i)}\{EFT(T_p) + TT(T_p,T_i)\}$;
        /*node that is normal critical parent of $T_i$ according [6]*/
10.     if($T_j$=CiriticalParent$_{T_i}$and $CP_{T_i} \neq CP_{T_j}$)
11.  $DOD(T_i,T_j) = (EST(T_j) + MET(T_j) + TT(T_i,T_j)) – (EST(T_3) +$
                $MET(T_3) + TT(T_i,T_3))$;
12.  else
13.  $DOD(T_i,T_j)=0$;
14.  end if
15.  end for
16.  for all ($CP_i, CP_j \in$ CriticalPaths)do
17.  $DOD(CP_i,CP_j)=\sum_{T_m \in T'_{CP_i}} \sum_{T_n \in T'_{CP_j}} DOD(T_m,T_n)+$
                $\sum_{T_n \in T'_{CP_j}} \sum_{T_m \in T'_{CP_i}} DOD(T_n,T_m)$
18.  end for
```

5. Performance Evaluation

In this section, experiments to verify the performance of the algorithm are proposed. The proposed algorithm is examined from two aspects:

a. Specifying the criteria for evaluating the quality of the proposed algorithm as compared with the other existing algorithms. For this purpose, two measures are used to compare quality: 1) Compare the shortest execution time of the algorithms, 2) Compare the financial cost of scheduling and the cheapest possible scheduling of each algorithm.

b. Evaluate the performance of the algorithm about workflows in comparison with the other existing algorithms. For this purpose, several workflows are studied.

5.1. Experimental Setup

For each experiment, we assume 10 clouds with each cloud having 10 separate services with different processor speeds and different prices. The processor speeds are selected randomly so that the fastest service is roughly five times faster than the slowest one, and accordingly, it is roughly five times more expensive. The average bandwidth between the computation services is set to 1 Gbps and the average bandwidth between clouds of 100 Mbps has been assumed. One-hour time slots are used. In the experiments, normal cost is calculated as follows:

$$NC=\frac{\text{total schedule cost}}{Cc} \quad (11)$$

where Cc is the cost of executing the same workflow with the cheapest strategy (scheduling of all nodes on a separate version of the cheapest available service) and normal makespan or execution time is calculated as follow:

$$NM =\frac{\text{schedule makespan}}{M_H} \quad (12)$$

whereM_H is the time of executing the same workflow with the fastest strategy (scheduling of all nodes on a separate version of the fastest available service). The final deadline is changed by a factor of α in the experiments. In this case, the deadline is calculated from the product of αat execution time of the fastest strategy.

5.2. Experimental Results

The proposed algorithm is tested for three common workflows presented in Table 2. We compared the proposed algorithms with IC-LOSS scheduling algorithm described in Section II because this algorithm has the same criteria as the proposed algorithm. For the Sipht, CyberShake, and Epigenomics workflows the proposed algorithm has a better performance, as shown in (Fig. 1, 2 and3). Experiments show that the proposed algorithm has a better performance than others when the workflow tasks have more data dependency and dispersal communication between them.

Table 2. Workflows Classes

Workflow	**Small**	**Medium**	**Large**
CyberShake	30-50	100	1000
Sipht	30-60	100	1000
Epigenimics	24-46	100	997

Algorithm 3: Extended PCP

```
19. Input:W=(T,E), T=$\cup_{i=1}^{n} T_i$ , E={($T_i$,$T_j$, $Data_{ij}$)|($T_i$,$T_j$)∈ T × T},
        $E_{ij}$=(Ti, Tj, dataij), CriticalPathes=$\cup_{i=1}^{s} CP_i$;
20. Output:$\cup_{i=1}^{n}\{(T_i, sched(T_i)\}$,workflow schedule;
21. ranks=ComputeRank(T);/*the ranks are calculated in a bottom-up
        direction in workflow*/
22. ST ←Sort(T, ranks);                /*Sort all T in descending ranks
        order*/
23. for all (Ti∈ST) do
24.  bestCloud ← ∅,T′ ← ∅,T″ ← ∅;   /*set them to empty*/
25.    $CP_{T_i}$ ← FindCriticalPath($T_i$);   /*Path of $T_i$ form CriticalPaths*/
26. T'←FindScheduleNodes($CP_{T_i}$); /*Nodes of  $CP_{T_i}$ that is scheduled*/
27.    for all(CP ⊂ CriticalPaths |DOD($CP_{T_i}$,CP)≠ 0) do /*T" is
        schedueledNodes in all depended on paths of $CP_{T_i}$*/
28. T"←T"∪FindScheduleNodes(CP);/*add Nodes of CP that is
        schedueled to T"*/
29.   end for
30. if(T'≠ ∅) then
31. bestCloud←FindBestCloudInPath($T_i$); /*bestCloud for $T_i$ according
        $CP_{T_i}$*/
32. else  if(T"≠ ∅)then
33. bestCloud←FindBestCloudInDependedPath($T_i$); /*Best Cloud for $T_i$
        According DependedPaths*/
34. else
35. bestCloud←PreCloudAssign() ;/*Estimate bestcloud  */
36. end if
37.    end if
38.    if(bestCloud≠ ∅andbestcloud is not full)then/*if bestcloud can
        accept more requests according to its strategy*/
39. AssignBestInstance($T_i$ ,bestCloud);/*Assign $T_i$ to BestInstance in
        bestCloud*/
40. else
41. AssignBestInstanceInFree($T_i$);/*Assign $T_i$ to BestInstance that exists
        in all Clouds*/
42. end if
43.   Mark $T_i$ as scheduledNode;
44. end for
```

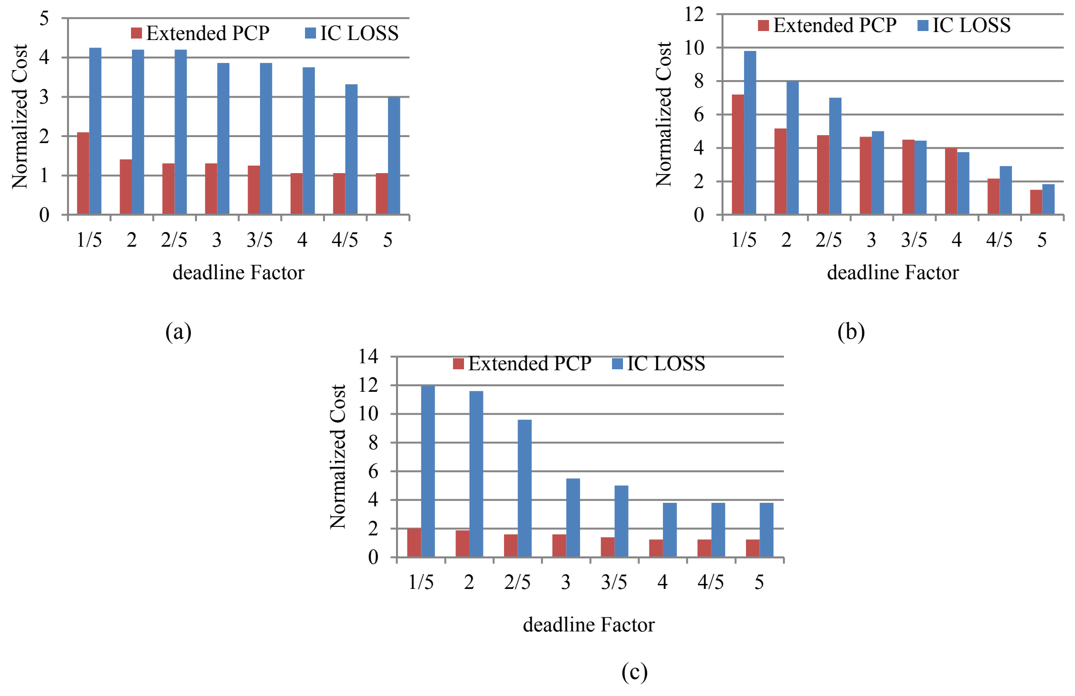
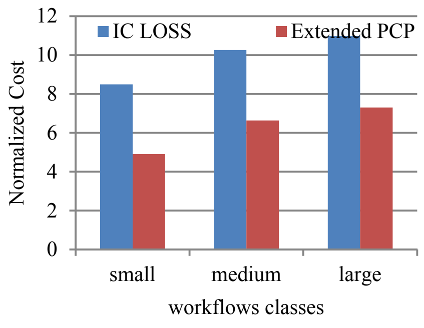
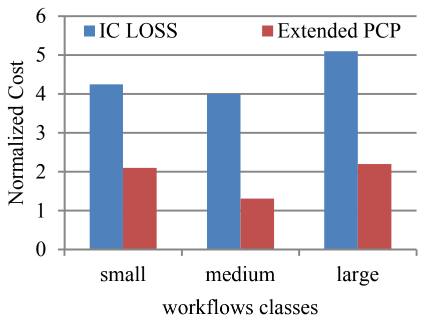


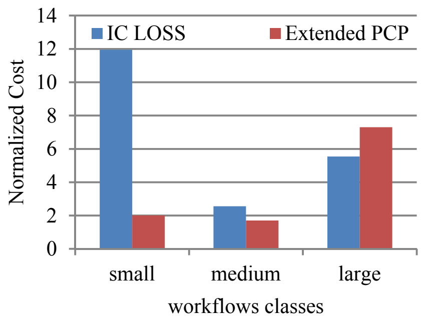
Fig. 1. The Normalized Cost of scheduling workflows with Extended PCP and IC-LOSS with the time interval equal to 1 h.
(a). CyberShake. (b).Sipht. (c). Epigenomics.



(a)

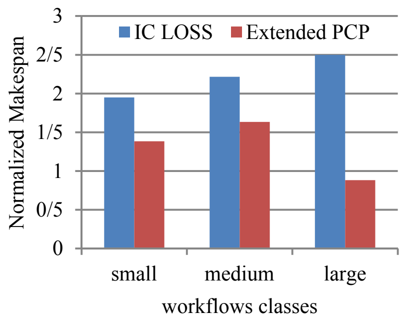


(b)

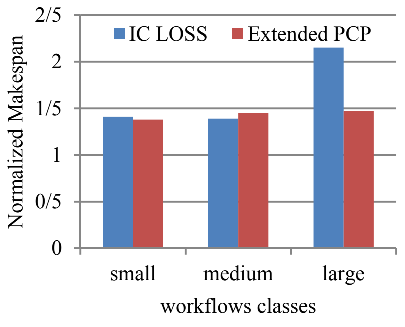


(c)

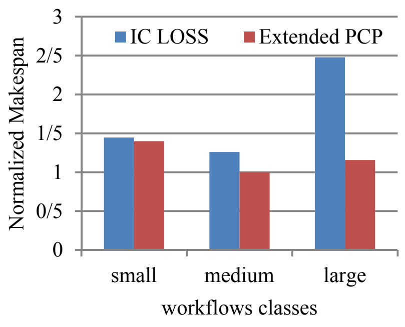
Fig. 2. The Normalized Cost of scheduling workflows with Extended PCP and IC-LOSS for different workflows classes. (a). CyberShake. (b). Sipht. (c). Epigenomics.



(a)



(b)



(c)

Fig. 3. The Normalized Makespan of scheduling workflows with Extended PCP and IC-LOSS for different workflows classes.(a). CyberShake. (b). Sipht. (c). Epigenomics.

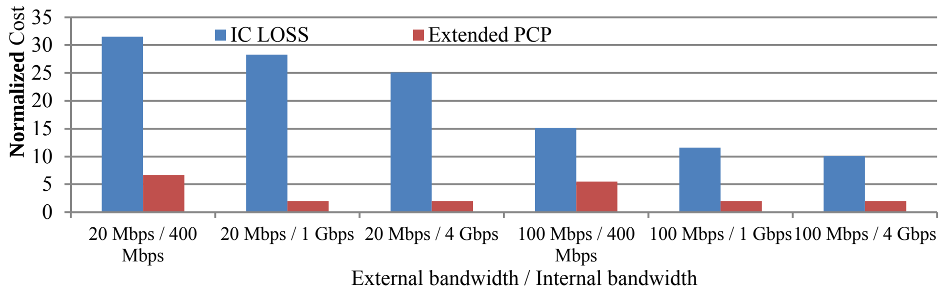
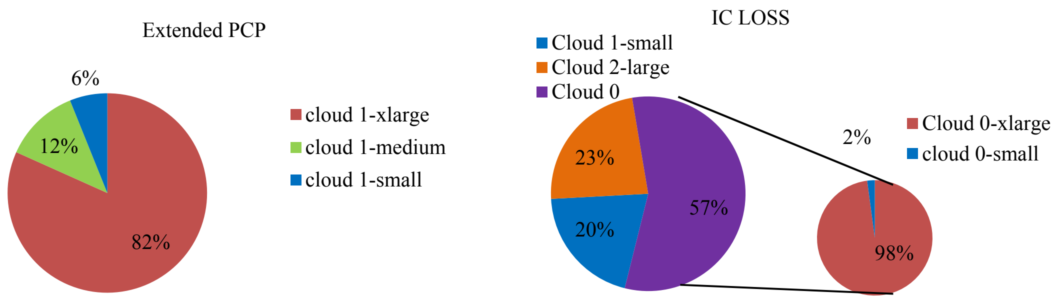
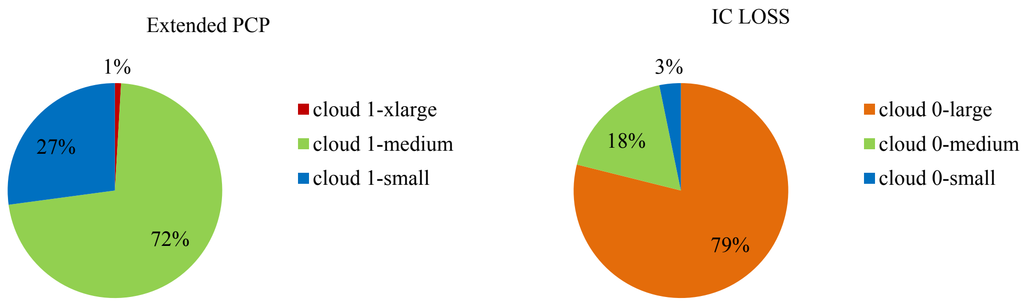


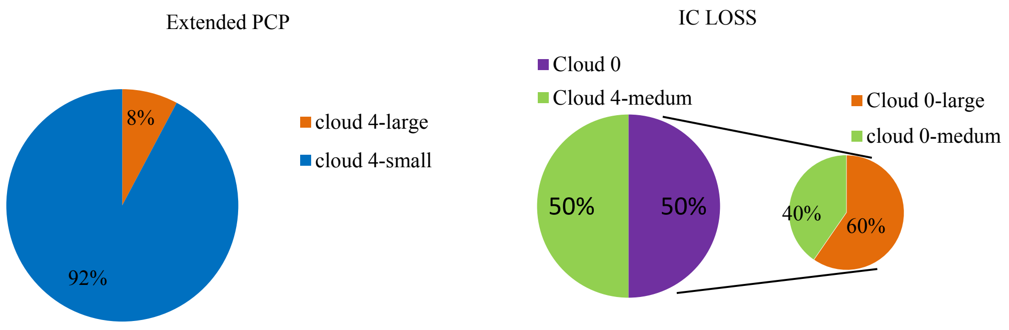
Fig. 4. The Normalized Cost of scheduling Epigenomics workflow for different external/internal bandwidth values



(a)



(b)



(c)

Fig. 5. Used instances (percentage of each instance type. a) CyberShake. b) Sipht. c) Epigenomics.

Fig 4 shows the Normalized Cost of scheduling Epigenomics workflow for different external/internal bandwidth values. As seen in Fig 4, the proposed algorithm shows more reaction to internal bandwidth and the cost is reduced by increasing it. Fig5. Shows the percentage of each instance type which is used by IC-Loss and Extended PCP algorithms. We can see from Fig.5 that the proposed algorithm tries to use the instances from one cloud, while IC-LOSS uses instances from different clouds. It leads to minimize the amount of communication between different clouds in our algorithm and so the cost and execution time are minimized.

6. Conclusions and Future Work

In this paper, a scheduling algorithm for data-driven workflows in a multi-cloud environment, called Extended PCP is proposed. In this proposed algorithm, we break a workflow into the paths whose tasks have the largest data exchange together and have the minimum data exchange between each other. A new criterion for controlling data flow, called Degree of Dependence or DOD, which is calculated for all nodes and paths is defined. This criterion will help us minimize the amount of communication between different paths by selecting the tasks that are more appropriate to be with each other. This algorithm considers communication as a very important factor that influences the whole scheduling. This makes the algorithm suitable for data-driven workflows in which a large amount of data is transferred between their tasks. The time complexity of the algorithms is $O(n^2)$, where n is the number of workflow tasks. The polynomial time complexity makes it a suitable option for the cases with large workflows. The proposed algorithm is evaluated by comparing its performance on scheduling three synthetic workflows that are based on real scientific workflows with different structures and different sizes, with IC-Loss [12]. The results show that the proposed algorithm has a better performance in workflows with a high amount of communication.

In the future, we intend to improve the proposed algorithm for multi-workflows scheduling in a multi-cloud environment and by taking into account other criteria such as fairness between providers. The next planned future work is extending the algorithm for dynamic environments where there is a possibility of changing the conditions of providers at any time in order to maximize the profits of users and providers at the same time.

References

[1] Petcu, D.: "Consuming resources and services from multiple clouds", *Journal of Grid Computing*, 12, (2), pp. 321-345, 2014.

[2] Yu, J., and Buyya, R.: "A taxonomy of scientific workflow systems for grid computing", *ACM Sigmod Record*, 34, (3), pp. 44-49,2005.

[3] Chen, R., Yang, M., Weng, X., Choi, B., He, B., and Li, X.: "Improving large graph processing on partitioned graphs in the cloud", in Editor (Ed.)^(Eds.): "Book Improving large graph processing on partitioned graphs in the cloud", (ACM, 2012, edn.), pp. 3.

[4] Li, A., Yang, X., Kandula, S., and Zhang, M.: "CloudCmp: comparing public cloud providers", in Editor (Ed.)^(Eds.): "Book CloudCmp: comparing public cloud providers" (ACM, edn.), pp. 1-14, 2010.

[5] Wang, G., and Ng, T.E.: "The impact of virtualization on network performance of amazon ec2 data center", in Editor (Ed.)^(Eds.): "Book The impact of virtualization on network performance of amazon ec2 data center" (IEEE, edn.), pp. 1-9, 2010.

[6] Abrishami, S., Naghibzadeh, M., and Epema, D.H.: "Deadline-constrained workflow scheduling algorithms for Infrastructure as a Service Clouds", Future Generation Computer Systems, 29, (1), pp. 158-169, 2013.

[7] Wu, Z., Ni, Z., Gu, L., and Liu, X.: "A revised discrete particle swarm optimization for cloud workflow scheduling", in Editor (Ed.)^(Eds.): "Book A revised discrete particle swarm optimization for cloud workflow scheduling" (IEEE, edn.), pp. 184-188, 2010.

[8] Pandey, S., Wu, L., Guru, S.M., and Buyya, R.: "A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments", in Editor (Ed.)^(Eds.): "Book A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments" (IEEE, edn.), pp. 400-407, 2010.

[9] Yu, J., and Buyya, R.: "Scheduling scientific workflow applications with deadline and budget constraints using genetic algorithms", *Scientific Programming*, 14, (3-4), pp. 217-230, 2006.

[10] Genez, T.A., Bittencourt, L.F., and Madeira, E.R.: "Workflow scheduling for SaaS/PaaS cloud providers considering two SLA levels", in Editor (Ed.)^(Eds.): "Book Workflow scheduling for SaaS/PaaS cloud providers considering two SLA levels" (IEEE, edn.), pp. 906-912, 2012.

[11] Bittencourt, L.F., and Madeira, E.R.M.: "HCOC: a cost optimization algorithm for workflow scheduling in hybrid clouds", *Journal of Internet Services and Applications*, 2, (3), pp. 207-227, 2011.

[12] Sakellariou, R., Zhao, H., Tsiakkouri, E., and Dikaiakos, M.D.: "Scheduling workflows with budget constraints","Integrated research in GRID computing", pp. 189-202, Springer, 2007.

[13] Topcuoglu, H., Hariri, S., and Wu, M.-y.,"Performance-effective and low-complexity task scheduling for heterogeneous computing", *Parallel and Distributed Systems,* IEEE Transactions on, 13, (3), pp. 260-274, 2002.

[14] Van den Bossche, R., Vanmechelen, K., and Broeckhove, J.,"Cost-optimal scheduling in hybrid iaas clouds for deadline constrained workloads", in Editor (Ed.)^(Eds.): "Book Cost-optimal scheduling in hybrid iaas clouds for deadline constrained workloads" (IEEE, edn.), pp. 228-235, 2010.

[15] Houidi, I., Mechtri, M., Louati, W., and Zeghlache, D.,"Cloud service delivery across multiple cloud platforms", in Editor (Ed.)^(Eds.),"Book Cloud service delivery across multiple cloud platforms" (IEEE, edn.), pp. 741-742, 2011.

[16] Li, W., Tordsson, J., and Elmroth, E.,"Modeling for dynamic cloud scheduling via migration of virtual machines", in Editor (Ed.)^(Eds.),"Book Modeling for dynamic cloud scheduling via migration of virtual machines" (IEEE, edn.), pp. 163-171, 2011.

[17] Fard, H.M., Prodan, R., and Fahringer, T.,"A truthful dynamic workflow scheduling mechanism for commercial multicloud environments", *Parallel and Distributed Systems,* IEEE Transactions on, 24, (6), pp. 1203-1212, 2013.

[18] Fard, H.M., Prodan, R., Barrionuevo, J.J.D., and Fahringer, T.,"A multi-objective approach for workflow scheduling in heterogeneous environments", in Editor (Ed.)^(Eds.),"Book A multi-objective approach for workflow scheduling in heterogeneous environments" (IEEE Computer Society, edn.), pp. 300-309, 2012.

[19] Duan, R., Prodan, R., and Li, X.,"Multi-objective game theoretic schedulingof bag-of-tasks workflows on hybrid clouds", *Cloud Computing,* IEEE Transactions on, 2, (1), pp. 29-42, 2014.

[20] Montes, J.D., Zou, M., Singh, R., Tao, S., and Parashar, M.,"Data-driven workflows in multi-cloud marketplaces", in Editor (Ed.)^(Eds.),"Book Data-driven workflows in multi-cloud marketplaces" (IEEE, edn.), pp. 168-175, 2014.